```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
using GilVerbekeTest.Models;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Table;
using System.Collections.Generic;

namespace GilVerbekeTest
{
    public static class Function1
    {
        [FunctionName("AddUserdata")]
        public static async Task<IActionResult> AddUserdata(
            [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = "userdata")]
HttpRequest req,
            ILogger log)
        {
            try
            {
                string requestbody = await new StreamReader(req.Body).ReadToEndAsync();
                Userdata reg = JsonConvert.DeserializeObject<Userdata>(requestbody);

                reg.UserdataId = Guid.NewGuid().ToString();

                string connectionstring =
Environment.GetEnvironmentVariable("AzureStorage");

                CloudStorageAccount cloudStorageAccount =
CloudStorageAccount.Parse(connectionstring);
                CloudTableClient cloudTableClient =
cloudStorageAccount.CreateCloudTableClient();
                CloudTable cloudTable = cloudTableClient.GetTableReference("userdata2");
                await cloudTable.CreateIfNotExistsAsync();

                UserdataEntity userdataEntity = new UserdataEntity(reg.Code,
reg.UserdataId)
                {
                    Code = reg.Code,
                    Age = reg.Age,
                    Gender = reg.Gender,
                    Province_geb = reg.Province_geb,
                    Studies = reg.Studies,
                    Headset = reg.Headset,
                    Province_opg = reg.Province_opg,
                    Studies_inst = reg.Studies_inst,
                    Hearing = reg.Hearing,
                    Taak1 = reg.Taak1,
                    Taak2 = reg.Taak2
                };

                TableOperation insertOperation = TableOperation.Insert(userdataEntity);
```

```
                await cloudTable.ExecuteAsync(insertOperation);
                return new OkObjectResult(reg);
            }
            catch (Exception ex)
            {
                log.LogError(ex.Message);
                return new StatusCodeResult(500);
            }
        }

        [FunctionName("AddTestdata")]
        public static async Task<IActionResult> AddTestdata(
            [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = "testdata")]
HttpRequest req,
            ILogger log)
        {
            try
            {
                string requestbody = await new StreamReader(req.Body).ReadToEndAsync();
                List<Testdata> testdatas =
JsonConvert.DeserializeObject<List<Testdata>>(requestbody);
                foreach (Testdata testdata in testdatas)
                {
                    Testdata reg = testdata;

                    reg.TestdataId = Guid.NewGuid().ToString();

                    string connectionstring =
Environment.GetEnvironmentVariable("AzureStorage");

                    CloudStorageAccount cloudStorageAccount =
CloudStorageAccount.Parse(connectionstring);
                    CloudTableClient cloudTableClient =
cloudStorageAccount.CreateCloudTableClient();
                    CloudTable cloudTable =
cloudTableClient.GetTableReference("testdata2");
                    await cloudTable.CreateIfNotExistsAsync();

                    TestdataEntity testdataEntity = new TestdataEntity(reg.Code,
reg.TestdataId)
                    {
                        Code = reg.Code,
                        Condition = reg.Condition,
                        Key_press = reg.Key_press,
                        Key_press_letter = reg.Key_press_letter,
                        Rt = reg.Rt,
                        Target = reg.Target,
                        Time_elapsed = reg.Time_elapsed,
                        Probe = reg.Probe,
                        Trial_part = reg.Trial_part,
                        Trial_type = reg.Trial_type,
                        Trial_index = reg.Trial_index,
                        Stimulus = reg.Stimulus,
                        Internal_node_id = reg.Internal_node_id,
                        Iteration = reg.Iteration,
                        Trial_version = reg.Trial_version

                    };
```

```
                    TableOperation insertOperation =
TableOperation.Insert(testdataEntity);
                    await cloudTable.ExecuteAsync(insertOperation);
                }
                return new OkObjectResult(testdatas);
            }
            catch (Exception ex)
            {
                log.LogError(ex.Message);
                return new StatusCodeResult(500);
            }
        }

        [FunctionName("AddBrowserdata")]
        public static async Task<IActionResult> AddBrowserdata(
            [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = "browserdata")]
HttpRequest req,
            ILogger log)
        {
            try
            {
                string requestbody = await new StreamReader(req.Body).ReadToEndAsync();
                List<Browserdata> browserdatas =
JsonConvert.DeserializeObject<List<Browserdata>>(requestbody);
                foreach (Browserdata browserdata in browserdatas)
                {
                    Browserdata reg = browserdata;

                    reg.BrowserdataId = Guid.NewGuid().ToString();

                    string connectionstring =
Environment.GetEnvironmentVariable("AzureStorage");

                    CloudStorageAccount cloudStorageAccount =
CloudStorageAccount.Parse(connectionstring);
                    CloudTableClient cloudTableClient =
cloudStorageAccount.CreateCloudTableClient();
                    CloudTable cloudTable =
cloudTableClient.GetTableReference("browserdata");
                    await cloudTable.CreateIfNotExistsAsync();

                    BrowserdataEntity browserdataEntity = new BrowserdataEntity(reg.Code,
reg.BrowserdataId)
                    {
                        Code = reg.Code,
                        Event = reg.Event,
                        Trial = reg.Trial,
                        Time = reg.Time
                    };

                    TableOperation insertOperation =
TableOperation.Insert(browserdataEntity);
                    await cloudTable.ExecuteAsync(insertOperation);
                }
                return new OkObjectResult(browserdatas);
            }
            catch (Exception ex)
```

```
            {
                log.LogError(ex.Message);
                return new StatusCodeResult(500);
            }
        }

        [FunctionName("GetTrialVersions")]
        public static async Task<IActionResult> GetTrialVersions(
            [HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = "trialversion")]
HttpRequest req,
            ILogger log)
        {
            try
            {
                List<Userdata> userdata = new List<Userdata>();
                string connectionString =
Environment.GetEnvironmentVariable("AzureStorage");

                CloudStorageAccount cloudStorageAccount =
CloudStorageAccount.Parse(connectionString);
                CloudTableClient cloudTableClient =
cloudStorageAccount.CreateCloudTableClient();
                CloudTable cloudTable = cloudTableClient.GetTableReference("userdata2");
                await cloudTable.CreateIfNotExistsAsync();

                TableQuery<UserdataEntity> query = new TableQuery<UserdataEntity>();

                var queryResult = await
cloudTable.ExecuteQuerySegmentedAsync<UserdataEntity>(query, null);
                var trialCount = new TrialCount();

                foreach (var row in queryResult.Results)
                {
                    if (row.Taak1 == "LDT1" && row.Taak2 == "PCT1")
                    {
                        trialCount.LDT1PCT1++;
                    }else if (row.Taak1 == "LDT2" && row.Taak2 == "PCT1")
                    {
                        trialCount.LDT2PCT1++;
                    }else if (row.Taak1 == "LDT1" && row.Taak2 == "PCT2")
                    {
                        trialCount.LDT1PCT2++;
                    }else if (row.Taak1 == "LDT2" && row.Taak2 == "PCT2")
                    {
                        trialCount.LDT2PCT2++;
                    }
                }

                var trialVersions = new TrialVersions();

                if (trialCount.LDT1PCT1 <= trialCount.LDT1PCT2 && trialCount.LDT1PCT1 <=
trialCount.LDT2PCT1 && trialCount.LDT1PCT1 <= trialCount.LDT2PCT2)
                {
                    trialVersions.Taak1 = 0;
                    trialVersions.Taak2 = 0;
                }else if (trialCount.LDT2PCT1 <= trialCount.LDT1PCT1 &&
trialCount.LDT2PCT1 <= trialCount.LDT1PCT2 && trialCount.LDT2PCT1 <= trialCount.LDT2PCT2)
                {
```

```
                    trialVersions.Taak1 = 1;
                    trialVersions.Taak2 = 0;
                }
                else if (trialCount.LDT1PCT2 <= trialCount.LDT1PCT1 && trialCount.LDT1PCT2
<= trialCount.LDT2PCT1 && trialCount.LDT1PCT2 <= trialCount.LDT2PCT2)
                {
                    trialVersions.Taak1 = 0;
                    trialVersions.Taak2 = 1;
                }
                else if (trialCount.LDT2PCT2 <= trialCount.LDT1PCT1 && trialCount.LDT2PCT2
<= trialCount.LDT1PCT2 && trialCount.LDT2PCT2 <= trialCount.LDT2PCT1)
                {
                    trialVersions.Taak1 = 1;
                    trialVersions.Taak2 = 1;
                }

                return new OkObjectResult(trialVersions);
            }
            catch (Exception ex)
            {
                log.LogError(ex.Message);
                return new StatusCodeResult(500);
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    class Browserdata
    {   public string BrowserdataId { get; set; }
        public string Code { get; set; }
        public string Event { get; set; }
        public int Trial { get; set; }
        public int Time { get; set; }
    }
}
```

```
using Microsoft.WindowsAzure.Storage.Table;
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    public class BrowserdataEntity : TableEntity
    {
        public BrowserdataEntity()
        {

        }
        public BrowserdataEntity(string code, string browserinteractionId)
        {
            this.PartitionKey = code;
            this.RowKey = browserinteractionId;
        }
        public string BrowserdataId { get; set; }
        public string Code { get; set; }
        public string Event { get; set; }
        public int Trial { get; set; }
        public int Time { get; set; }
    }
}
```

```cs
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    class Testdata
    {
        public string TestdataId { get; set; }
        public string Code { get; set; }
        public string Condition { get; set; }
        public int? Key_press { get; set; }
        public string Key_press_letter { get; set; }
        public double? Rt { get; set; }
        public string Target { get; set; }
        public int Time_elapsed { get; set; }
        public string Probe { get; set; }
        public string Trial_part { get; set; }
        public string Trial_type { get; set; }
        public int Trial_index { get; set; }
        public string Stimulus { get; set; }
        public string Internal_node_id { get; set; }
        public double? Iteration { get; set; }
        public string Trial_version { get; set; }
    }
}
```

```
using Microsoft.WindowsAzure.Storage.Table;
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    public class TestdataEntity : TableEntity
    {
        public TestdataEntity()
        {

        }
        public TestdataEntity(string code, string userdataId)
        {
            this.PartitionKey = code;
            this.RowKey = userdataId;
        }
        public string TestdataId { get; set; }
        public string Code { get; set; }
        public string Condition { get; set; }
        public int? Key_press { get; set; }
        public string Key_press_letter { get; set; }
        public double? Rt { get; set; }
        public string Target { get; set; }
        public int Time_elapsed { get; set; }
        public string Probe { get; set; }
        public string Trial_part { get; set; }
        public string Trial_type { get; set; }
        public int Trial_index { get; set; }
        public string Stimulus { get; set; }
        public string Internal_node_id { get; set; }
        public double? Iteration { get; set; }
        public string Trial_version { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    class TrialCount
    {
        public int LDT1PCT1 { get; set; }
        public int LDT2PCT1 { get; set; }
        public int LDT1PCT2 { get; set; }
        public int LDT2PCT2 { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    class TrialVersions
    {
        public int Taak1 { get; set; }
        public int Taak2 { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    class Userdata
    {
        public string UserdataId { get; set; }
        public string Code { get; set; }
        public int Age { get; set; }
        public string Gender { get; set; }
        public string Province_geb { get; set; }
        public string Studies { get; set; }
        public string Headset { get; set; }
        public string Province_opg { get; set; }
        public string Studies_inst { get; set; }
        public string Hearing { get; set; }
        public string Taak1 { get; set; }
        public string Taak2 { get; set; }
    }
}
```

```
using Microsoft.WindowsAzure.Storage.Table;
using System;
using System.Collections.Generic;
using System.Text;

namespace GilVerbekeTest.Models
{
    public class UserdataEntity : TableEntity
    {
        public UserdataEntity()
        {

        }
        public UserdataEntity(string code, string userdataId)
        {
            this.PartitionKey = code;
            this.RowKey = userdataId;
        }
        public string UserdataId { get; set; }
        public string Code { get; set; }
        public int Age { get; set; }
        public string Gender { get; set; }
        public string Province_geb { get; set; }
        public string Studies { get; set; }
        public string Headset { get; set; }
        public string Province_opg { get; set; }
        public string Studies_inst { get; set; }
        public string Hearing { get; set; }
        public string Taak1 { get; set; }
        public string Taak2 { get; set; }
    }
}
```